APPENDIX D

TRIM.FaTE COMPUTER FRAMEWORK

[This page intentionally left blank.]

APPENDIX D
TRIM.FaTE COMPUTER FRAMEWORK

The TRIM.FaTE computer framework provides the infrastructure required to conduct and analyze TRIM.FaTE simulations. The framework allow users to:

- Define the issue to be studied, including time period, geographic region, pollutants, media, and populations of interest;
- Specify and choose algorithms that will be used for simulations;
- Specify modeling parameters, including emissions sources, characteristics of the environment (*e.g.*, air temperature and soil permeability), and simulation time step;
- Identify data sets to be used and created;
- Execute the simulation;
- Perform sensitivity and Monte Carlo studies; and
- Export and process results.

The development of the TRIM.FaTE framework began with the creation of a series of prototypes. These prototypes served as test beds for evaluating approaches and allowed changes to be quickly implemented and tested. The lessons learned from Prototypes I through V were incorporated into Version 1.0 of the TRIM.FaTE framework with the addition of features that increase the usefulness of the system, such as management of multiple modeling scenarios, portability between Windows and UNIX, and improved ease of use and robustness. The Version 1.0 framework has served as the basis for all subsequent versions. The current version, Version 2.5, was released in July 2002 and includes expanded analysis tools and stochastic modeling capabilities.

This description of the TRIM.FaTE computer framework generally covers both the prototypes and Version 2.5 with indications where necessary that descriptions apply to only one of the implementations. As the software architecture and implementation for Version 1.0 and all subsequent versions are very similar, only the Version 2.5 will be described in this chapter. Additional information about the architecture and design of TRIM.FaTE Version 1.0 can be found in Fine et al. (1998a,b).

D.1 SOFTWARE ARCHITECTURE

Bass et al. (1998) provide the following definition of software architecture:

> The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them.

As the prototypes and Version 2.5 have different architectures, they are described separately in this section.

D.1.1 ARCHITECTURE OF THE PROTOTYPES

The prototypes were implemented in an object-oriented manner, with almost all important quantities implemented as objects/classes. These include:

- parcels;
- volume elements;
- compartments;
- chemicals;
- links;
- algorithms;
- parameters (input parameters and calculated parameters);
- runs; and
- projects.

In the prototypes, a project was constructed in a hierarchical fashion: first a parcel was created, then volume elements were added "to" the parcel, and then compartments were added to the volume elements. Links were created manually or automatically determined based on the spatial adjacency information of the project.

When a run was initiated, the needed transition matrices, source term vectors, and initial condition vector were constructed from the modeled system. This process utilized the link topology and algorithms associated with each link, in addition to the source specified for particular compartments and the implied source terms calculated based on any boundary air concentrations specified. The transition matrices and associated source term and initial condition vectors were used in successive calls to the differential equation solver (*i.e.,* LSODE), after which the predicted chemical mass in each compartment was available.

An expression evaluator was also included within the design of the prototypes. This is used to evaluate almost all algorithms and other needed calculated quantities (*e.g.*, distribution coefficients in soil for organics, which were calculated from properties of the chemical and the soil compartment). The expressions themselves were stored as strings, using an object-oriented syntax consistent with the overall object model used. These expressions were "compiled" when a run was performed, with the objects needed to calculate each expression obtained for subsequent calculation. This allowed flexible naming of variables and the creation of numerous

intermediate terms that helped provide insight into the finer details of a particular run. Further, it significantly improved the quality of output reports that could be produced. For example, detailed reports could be generated that showed the exact equations used to calculate a given quantity, as well as the values of the terms used in its calculation. The successful implementation of such a system in the prototype made it possible to seriously consider, and ultimately decide upon, implementing a similar capability in the TRIM.FaTE Version 1.0 and subsequent versions.

D.1.2 VERSION 2.5 ARCHITECTURE

As shown in Figure D-1, the TRIM.FaTE computer system architecture is complex yet flexible. The architecture components used to describe TRIM are classified as those that primarily provide (1) functionality (rectangles), and (2) those that primarily provide data (ovals). However, each of the components except for external data sources provide both functionality and data. The architectural components that are implemented to some degree in Version 2.5 are depicted with shadows. This figure is designed to represent the relationships within the TRIM computer framework, rather than the data flow within the system. Therefore, the word along an arrow forms a sentence where the verb on the arrow connects the two architecture components at the end of an arrow. For example, in the upper left hand corner of the figure, the TRIM Core "invokes" Analysis and Visualization Tools. Each of the TRIM components shown in Figure D-1 are described below. Note that this framework is used only for TRIM.FaTE. The remaining modules will be integrated together into EPA's Multimedia Integrated Modeling System (MIMS) (Fine et al. 2002).

D.1.2.1 TRIM.FaTE Core

The TRIM.FaTE Core primarily provides services required by multiple TRIM components or to integrate those components. The following functions are provided by the Core:

- Coordination of TRIM graphical user interface components. This includes allowing the user to invoke TRIM modules, such as TRIM.FaTE, and maintaining lists of open windows.

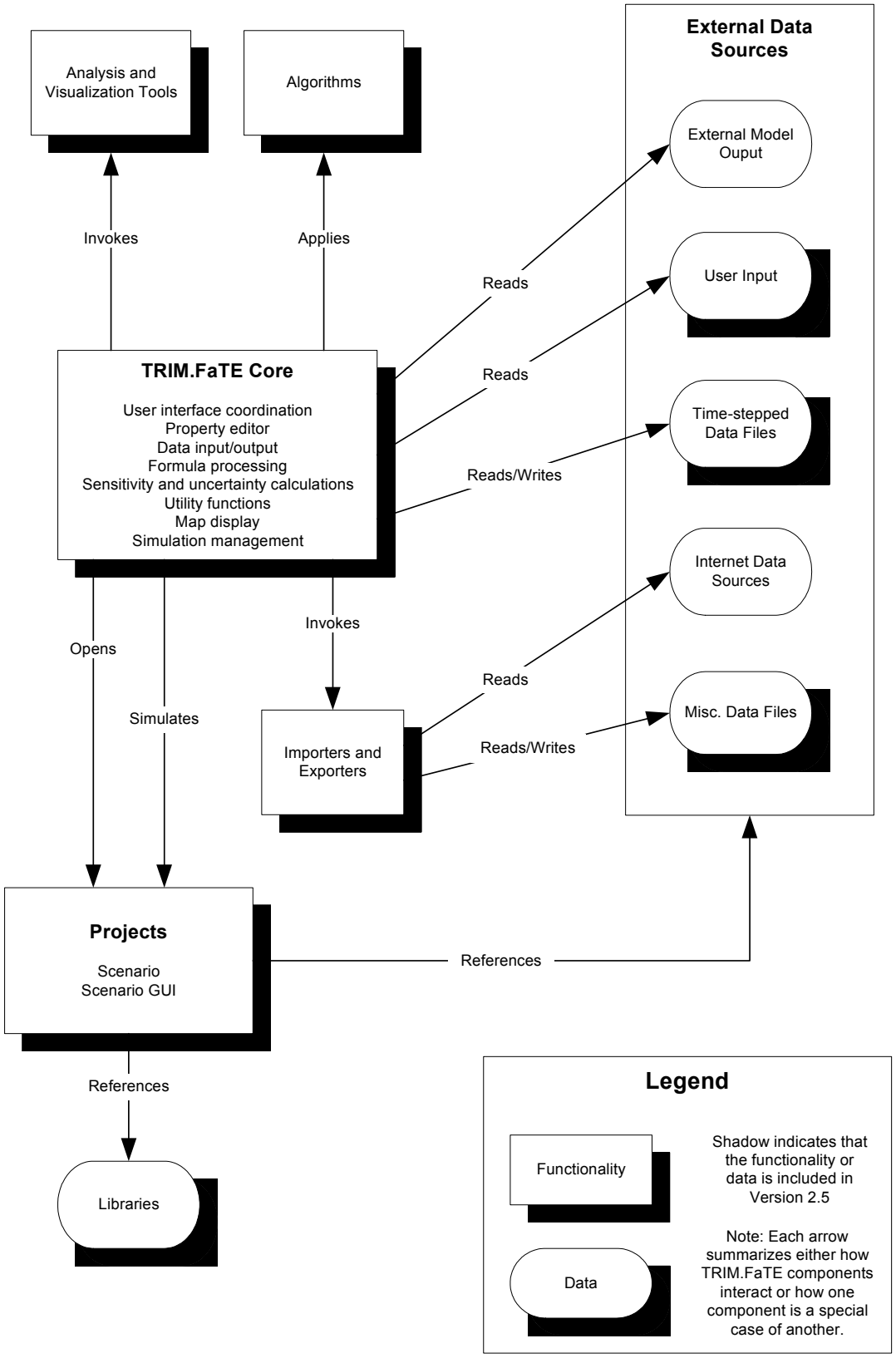
- Allowing users to edit and view property values, where a property is a parameter or attribute (*e.g.*, molecular weight) that describes an entity used by a model, such as a compartment or volume element. Properties include air temperature, scavenging coefficients, and chemical reaction rates.

- Interpretation of formulas used to describe property values. TRIM.FaTE allows users to write formulas using a simple language for conveying expressions, developed for TRIM.FaTE.

- Calculation of sensitivity and uncertainty.

- Utility functions used by TRIM.FaTE, such as routines to assist with common graphical user interface (GUI) and geometric operations.

Figure D-1
TRIM.FaTE Computer System Architecture



- Utility functions used by TRIM.FaTE, such as routines to assist with common graphical user interface (GUI) and geometric operations (*e.g.*, computing parcel overlaps and adjacencies, low-level routines used to represent points, lines, and polygons such as those used to define parcels and volume elements).

- Spatial presentation ("mapping") of TRIM.FaTE compartment results with respect to volume elements. Future versions will be able to display arbitrary supplemental information supplied by the user, such as soil types. The mapping tool will also allow users to specify the X-Y extent of TRIM.FaTE volume elements.

- Coordination of simulation functions, including management of algorithms that compute chemical transfer coefficients between and transformation coefficients within conceptual compartments and controls that start and stop simulations, display error messages, and manage complex sets of simulations.

D.1.2.2 Project

All information pertinent to an environmental study is stored in a "project." Each project is responsible for displaying the information it contains and allowing the user to change the information, in some cases relying on a TRIM.FaTE Core functionality such as the property editor. A project can contain one or more "scenarios," where each scenario contains a complete description of a model run (or set of related runs), a description of the outdoor environment being simulated, populations being studied, and model parameters, such as the simulation time step.

D.1.2.3 Libraries

A substantial amount of relatively static information is required to conduct studies of multimedia fate and transport and effects on selected populations. For instance, the measured properties of chemicals change infrequently. Also, the boundaries of a study region might stay constant for years. In addition, many of the characteristics of and relationships between compartments are defined using formulas that do not change for a given simulation. Users can store such information in TRIM object libraries. They can then easily reuse selected information from a library in future projects. When information from a library is used in a project, a copy is made of the information, which protects the project from future changes to the library.

D.1.2.4 External Data Sources, Importers, and Exporters

It will be common for TRIM.FaTE users to access or create data sets beyond TRIM.FaTE projects. Some data sets may be too large to be conveniently stored in projects, while other data sets already exist in non-TRIM formats. TRIM.FaTE provides several methods for accessing external information. The TRIM.FaTE Core accepts user inputs and will read and write data in files in the standard TRIM.FaTE format. These files are currently delimited ASCII text files that can be imported into spreadsheets. In the future, output directly to a database will be considered.

TRIM.FaTE is modular and allows new importers and exporters to be developed quickly and easily. Current data importers read data sets created outside of TRIM.FaTE, create TRIM.FaTE objects as needed, and set appropriate TRIM.FaTE properties. For example, an importer could read files containing measurements of surface air temperature and set properties in ground-level TRIM.FaTE air domains. Data exporters provide TRIM.FaTE results in a form that is suitable for use by another program or for interactive review. This could include comma-delimited files that could be imported into a spreadsheet and tabular results for people to review and HTML formatted files that allow the user to view links, transfer factors, and other information for a time step of a specific scenario.

D.1.2.5 Analysis and Visualization Tools

Version 2.5 includes several analysis and visualization tools. The first group of tools, which includes the Graphical Results Viewer and the Food Web Viewer, are used to display results and simulation setup details graphically. The second group of tools, which includes the Averager, the Aggregator, and the Transposer, are post-processing tools used to process TRIM.FaTE moles, mass, and concentration output files. In addition, simulation results can be easily exported to Excel or other analysis packages. In the future, TRIM.FaTE may include additional analysis and visualization capabilities.

D.2 IMPLEMENTATION APPROACHES

The computer framework has been developed using an object-oriented approach. There has been much discussion in the software engineering literature (*e.g.,* Booch 1993) on the benefits of this approach, including increased software extensibility, reuse, and maintainability. The essence of object-oriented software development is that concepts, such as a volume element, are represented as units that contain internal data (*e.g.*, the boundaries of a volume element) and operations on that data (*e.g.*, compute volume). Additionally, that one class of objects (*e.g.*, volume element with vertical sides) can be a specialization of another class of objects (*e.g.*, volume element). Being able to specialize classes of objects allows general functionality to be shared by several specialized classes. TRIM.FaTE's view of the environment (with volume elements that contain compartments) and the development of associated graphical user interfaces are well suited for an object-oriented treatment.

TRIM.FaTE is being developed in an iterative manner. The major components and responsibilities of a class of objects are understood before implementation, but some details are worked out as implementation proceeds. During implementation, the design is modified as needed. The object-oriented, open-ended structure of TRIM is intended to make future changes and additions a relatively simple process. For Version 2.5 of TRIM, simpler and/or more reliable approaches were used in preference to faster and/or less resource-intensive approaches. In cases where simple approaches will not have adequate performance or will significantly limit the potential for future changes, more complex approaches were used. As time permits, operations that cause noticeable speed or resource problems will be optimized.

D.3 IMPLEMENTATION LANGUAGE

Due to the different objectives of the framework prototypes and Version 2.5, different development languages were chosen. The rationale for each choice is described below.

D.3.1 PROTOTYPES

Microsoft's Visual Basic was used as the primary tool with which to implement the prototypes. This was due to a number of factors:

- Ease of use with Microsoft Excel, to which all members on the team had access (for early prototypes);
- Object-oriented features of language, while limited[1], simplify a dynamic, iterative architecture development cycle; and
- Straightforward to call needed Fortran codes (*e.g.*, differential equation solver, linear equation solver, triangulation).

D.3.2 VERSION 2.5

The Version 2.5 computer framework was developed primarily, but not entirely, in the Java programming language. Some parts of TRIM.FaTE, such as the differential equation solver, are implemented in FORTRAN, and other parts, such as the polygon overlay algorithm, are implemented in C.

Advantages of using Java include the following.

- Java code is portable across different hardware and operating systems. This is especially important for graphical user interfaces, which will comprise a large fraction of the TRIM code and which can be difficult to develop for multiple platforms.
- Java offers a good combination of speed of development, robustness, and support for object-oriented designs.
- Java is supported by multiple vendors. This often leads to competitive pressures to improve development tools, and it reduces the likelihood that one vendor's product strategy or financial problems will cripple TRIM development.
- Java provides built-in support for multithreading, which allows multiple operations to proceed simultaneously, and networking.

[1] The primary limitation is that Visual Basic does not support inheritance. However, it does support polymorphism (an object/class can implement any number of interfaces), which is utilized to a large degree to simplify the logic of the programming.

The disadvantages of using Java include the following:

- Java programs typically execute more slowly than programs written in C++ or BASIC. As the technologies for compiling and executing Java programs advance, the speed penalty for using Java should decrease.

- Fewer plug-in components (*e.g.*, mapping tools) and libraries (*e.g.*, matrix manipulation) are available for Java than there are for languages such as C++ or BASIC on Windows, but the number of Java components available is continuing to grow.

- Java development tools are not as mature as tools for other languages, but that situation is improving.

D.4 EMBEDDED FUNCTIONS

As described elsewhere, TRIM.FaTE allows users to specify and choose algorithms that compute chemical transformation and transfer factors. This provides significant flexibility to describe different pollutants and environmental systems. However, some transfer algorithms are too complex to be represented as user-entered formulas. These algorithms are described below.

D.4.1 ADVECTIVE TRANSPORT BETWEEN AIR COMPARTMENTS

The advective transport from one air compartment to another is calculated as the sum of the transport and dispersive/lateral wind speed. The methods used to calculate these quantities are implemented in subroutines within the source code, rather than through the use of expressions for the expression evaluator. Details on these methods can be found in Section 3.1 of the TRIM.FaTE TSD Volume II.

D.4.2 INTERFACIAL AREA BETWEEN VOLUME ELEMENTS

The interfacial area shared by volume elements is used frequently (*e.g.*, for advective and diffusive transfers). This is calculated by subroutines in the source code itself. In the prototype, each side of a volume element that might intersect another volume element is triangulated (in conjunction with a dynamic link library for triangulation). Next, intersection of the triangulations is computed. Version 2.5 uses a more specialized but faster approach that takes advantage of current restrictions on the structure of volume elements (sides must be vertical and tops and bottoms horizontal). The X-Y projections of side-by-side volume elements are examined for line segment overlap. The length of the overlap multiplies the extent of vertical overlap. The interfacial area for volume elements that are stacked vertically is computed by intersecting the polygons that represent the X-Y projections of the volume elements and then computing the area of the resulting polygon. When more general shapes are permitted for volume elements, a more general calculation, such as the triangulation approach, will be incorporated into Version 2.5.

D.4.3 AREA OF VOLUME ELEMENTS

The area of volume elements is also used frequently (e.g., for deposition and resuspension transfers). It is a simple calculation in the source code that uses the X-Y coordinates of a volume element to calculate the horizontal area of the volume element. This function is called in the volume element property "area."

D.4.4 VOLUME OF VOLUME ELEMENTS

The volume of volume elements is used in nearly every algorithm involving an abiotic compartment. It is a simple calculation in the source code that uses the X-Y coordinates and height of a volume element to calculate the volume of the volume element. This function is called in the volume element property "volume."

D.4.5 BOUNDARY CONTRIBUTION

This function computes the advective contribution from a boundary of the modeling system to an air compartment. It requires as inputs the boundary concentration (in grams per cubic meter), wind speed (in meters per second), and wind direction (in degrees clockwise from north). This function is called in the boundary air compartment property "boundaryContribution."

D.4.6 OTHER MATHEMATICAL FUNCTIONS

In addition to the TRIM-specific functions described above, TRIM.FaTE also includes the following standard mathematical functions that can be used by the user in writing algorithms and/or defining object properties.

- $abs(x_1)$ - absolute value of x_1
- $exp(x_1)$ - exponential of x_1
- $ln(x_1)$ - natural logarithm of x_1
- $log10(x_1)$ - log base 10 of x_1
- $max(x_1,x_2,x_3,...,x_n)$ - maximum of values $x_1...x_n$
- $min(x_1,x_2,x_3,...,x_n)$ - minimum of values $x_1...x_n$
- $mod(x_1,x_2)$ - modulo of x_1 relative to x_2
- $sqrt(x_1)$ - square root of x_1

D.5 REFERENCES

Bass, L., P. Clements, and R. Kazman. 1998. Software architecture in practice. Reading, MA: Addison-Wesley.

Booch, G. 1993. Object-oriented analysis and design with applications. Redwood City, California: The Benjamin/Cummings Publishing Company, Inc.

Fine, S.S., S.C. Howard, A.M. Eyth, D.A. Herington, and K.J. Castleton. 2002. The EPA Multimedia Integrated Modeling System Software Suite. Presentation at the Second Federal Interagency Hyrdologic Modeling Conference, July 28-August 1, Las Vegas, Nevada.

Fine, S.S., A. Eyth, and H. Karimi. 1998a. The Total Risk Integrated Methodology (TRIM) Computer System Architecture. Research Triangle Park, NC: MCNC-North Carolina Supercomputing Center. November.

Fine, S.S., A. Eyth, and H. Karimi. 1998b. The Total Risk Integrated Methodology (TRIM) Computer System Design. Research Triangle Park, NC: MCNC-North Carolina Supercomputing Center. November.